TECHNIKA TRANSPORTU SZYNOWEGO

Aldona KUŚMIŃSKA-FIJAŁKOWSKA, Zbigniew ŁUKASIK

PROGRAMOWANIE W JĘZYKU KAREL ROBOTA PRZEMYSŁOWEGO FANUC S-420F STRUKTURYZACJA PROGRAMU - PROCEDURY

Streszczenie

W artykule przedstawiono zrobotyzowany system w oparciu o Robota Fanuc S-420F., który umożliwia przećwiczenie praktycznych aspektów programowania robota przemysłowego przygotowywując do pracy w roli integratora zintegrowanych systemów przemysłowych.

WSTĘP

Język KAREL jest połączeniem elastyczności języków wysokiego poziomu z efektywnością języków sterowania maszynowego. W nowszych sterownikach serii R-J bardzo dużą część zadań języka KAREL przejął język TPE, którego programy tworzy się wprost za pomocą Teach Pendant'a [2]. Jednakże język ten jest stale rozwijany i obecny najnowszych sterownikach robotów FANUC[9]. Ze względu na elastyczność i poziom zaawansowania, który jest dostępny już w kontrolerze R-H, język ten jest niezastąpionym narzędziem przy tworzeniu dużych projektów i wyrafinowanych programów, które mogą pracować bezpośrednio z użyciem danych CAD'owskich oraz sterować dużą ilością robotów w tym samym czasie.

1. ZINTEGROWANY ZROBOTYZOWANY SYSTEM

Zintegrowany System w oparciu o robota przemysłowego S-420F firmy FANUC Robotics (**F**actory **A**utomation **NU**merucal **C**ontrol) przestawiono na Rys. 1.1.

Rys. 1.1. Zrobotyzowane Stanowisko a) Stanowisko robocze; b) Kontroler R-H; c) Skaner SICK S3000 [5]

W skład systemu wchodzą [5]:

1. Podsystem sterowania
   - kontroler typu R-H robota FANUC S-420F
2. Podsystem bezpieczeństwa
   - skaner Sick S3000,
   - bariery uniemożliwiające dostęp w obszar roboczy ramienia,
   - drzwi wejściowe zabezpieczone krańcówką mechaniczną,
   - system przycisków zatrzymania awaryjnego.
3. Podsystem oprzyrządowania złożony z:
   - ramienia roboczego robota FANUC S-420F (Rys. 1.1a),
   - mobilnej tablicy o regulowanym kącie pochylenia (Rys. 1.1b),
   - narzędzia roboczego, którym jest pisak umieszczony w uchwycie przymocowanym do czoła montażowego robota (Rys. 1.1c).

2. JĘZYK KAREL W PROGRAMOWANIU ROBOTÓW FANUC

Program w języku KAREL składa się z logiki programu oraz danych. Logikę programu stanowi sekwencja kroków, których wykonanie prowadzi do osiągnięcia określonego wyniku. Logika programu oraz dane są od siebie oddzielone. Logika programu umieszczona jest między słowami kluczowymi BEGIN i END. Dane programu to zmienne zadeklarowane w sekcji VAR przy pomocy: nazwy, typu oraz lokalizacji w odpowiednim obszarze pamięci. Wartości danych mogą być „nauczone” (zapamiętane), jako wartości obecnego położenia ramienia robota, mogą zostać obliczone przez program odczytane z plików danych lub wprowadzone przez operatora w trakcie wykonywania programu za pomocą Teach Penadant'a, CRT/KB albo innego urządzenia. Wartości danych mogą być zatem różne między kolejnymi wykonaniami tego samego programu. Dane i logika są odseparowane by:

- punkty i ścieżki punktów mogły być używane w kilku miejscach,
- wiele programów mogło korzystać z tych samych danych,
- ten sam program mógł korzystać z różnych zestawów danych,
- używać danych generowanych przez programy CAD-owskie.

Program w języku KAREL składa się z deklaracji oraz z kodu wykonywalnego przechowywanych w pliku źródłowym[4].. Wartości zmiennych związanych z programem przechowywane są w pliku zmiennych. Translator języka KAREL tłumaczy kod źródłowy na p-code i generuje odpowiedni plik. Jeśli program zostanie wybrany na liście KAREL'u, to plik p-code oraz plik zmiennych ładowane są automatycznie. System tworzy w tym momencie wszystkie wymagane zmienne, których nie było w pamięci, jako niezainicjowane. Gdy uruchamiany jest program, to KAREL wykonuje instrukcje p-code'u. Programy języka KAREL mogą być tworzone i edytowane na CRT/KB lub na innym komputerze wyposażonym w edytor języka KAREL, a następnie przesłane do kontrolera robota [7].

3. STRUKTURA PROGRAMU W JĘZYKU KAREL

Program w języku KAREL składa się z deklaracji oraz części wykonywalnych utworzonych z użyciem instrukcji i innych elementów języka KAREL. Poniżej przedstawiono ogólną strukturę programu w języku KAREL[7].

PROGRAM prog_name
CONST and/or VAR declarations with translator directives
ROUTINE declarations

BEGIN
executable statements
END prog_name

ROUTINE declarations

W powyższym przykładzie słowa napisane dużymi literami są słowami zarezerwowanymi języka KAREL o określonym znaczeniu. **PROGRAM**, **CONST**, **VAR** i **ROUTINE** wskazują sekcje deklaracyjne programu.

Każdy program w języku KAREL rozpoczyna instrukcja **PROGRAM**, po której następuje identyfikator będący nazwą programu. Identyfikator użyty jako nazwa programu po słowie kluczowym **PROGRAM** może zostać użyty jedynie jako oznaczenia końca sekcji wykonywalnej programu po słowie kluczowym **END**.

Kolejnymi elementami programu są sekcje deklaracyjne **CONST** – stałych i **VAR** zmiennych, które mogą zawierać dowolną ilość deklaracji. Mogą one się wzajemnie przeplatać i być wielokrotnie umieszczane w dowolnej kolejności. Dobrym zwyczajem ze względu na czytelność jest ich grupowanie. Sekcje te to dane programu. Wartości danych mogą być „nauczone” przez odczytanie obecnego położenia TCP, mogą zostać obliczone przez program lub wprowadzone przez użytkownika w trakcie wykonywania programu za pomocą Teach Penadant'a albo CRT/KB i ich wartości mogą być różne przy kolejnych uruchomieniach danego programu [7].

Po tej sekcji następuje obszar deklaracji procedur, których budowa podobna jest do budowy programu lecz każda z procedur musi zaczynać się od słowa kluczowego **ROUTINE**. Deklaracje procedur mogą występować także po części wykonywalnej programu.

Część wykonywalna programu umieszczona jest między słowami kluczowymi **BEGIN** oraz **END** i stanowi logikę programu. Logika ta jest sekwencją kroków, których wykonanie prowadzi do osiągnięcia określonego wyniku za pomocą wbudowanych w język struktur, operacji oraz procedur takich jak [7]:

- proste i strukturalne typy danych,
- operacje arytmetyczne, relacyjne i Boolowskie,

– pętle i instrukcje wyboru,
– procedury i funkcje,
– rozkazy kontroli ruchu,
– operacje wejścia i wyjścia.

4. LISTING PROGRAMU W JĘZYKU KAREL ZINTEGROWANEGO ZROBOTYZOWANEGO SYSTEMU

Napisany listing programu lab 1 dla Zintegrowanego Zrobotyzowanego Systemu ma na celu przedstawienie budowę przykładowego programu napisanego w języku KAREL, a także metody optymalizacji oraz poprzez użycie procedur zwiększanie czytelności kodu programu.

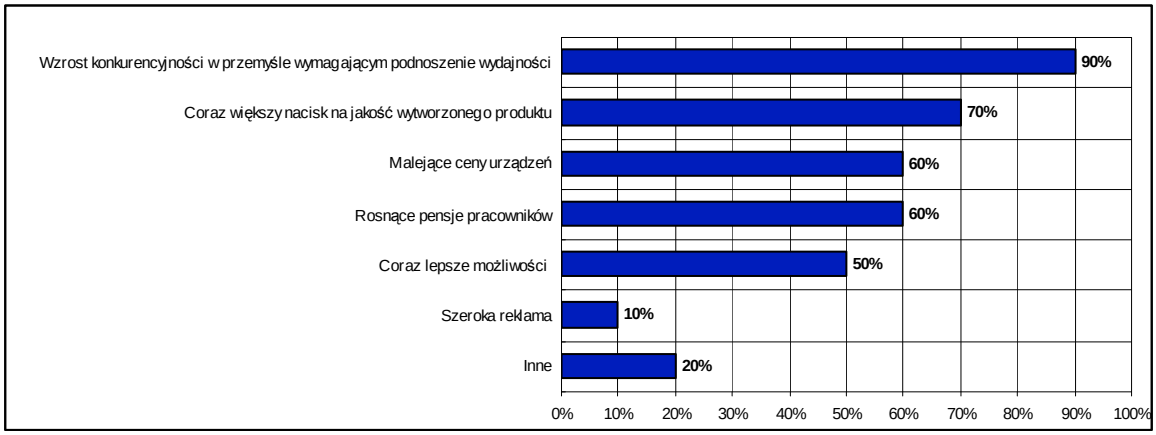
PROGRAM lab1 [7]

```
-- ********************************************************************** --
-- **                        Laboratorium 1                            ** --
-- ********************************************************************** --

VAR                           -- deklaracja zmiennych
  base_tool   ,               -- narzedzie bazowe - czolo montazowe
  base_frame  ,               -- bazowy uklad wspolrzednych
  table_frame ,               -- uklad wspolrzednych tablicy
  def_frame   ,               -- uklad definiowany za pomoca FRAME
  lab_tool    ,               -- uklad wspolrzednych narzedzia
  home        ,               -- punkt wyjsciowy - spoczynkowy
  start       ,               -- pozycja przejsciowa - bezpieczna
  clean_tool  ,               -- pozycja czyszczenia narzedzia
  pick        ,               -- pozycja pobrania detalu
  offset      ,               -- offset
  dest_pt1    ,               -- punkt docelowy 1
  dest_pt2    ,               -- punkt docelowy 2
  dest_pt3    ,               -- punkt docelowy 3
  dest_pt4    ,               -- punkt docelowy 4
  frm_orig    ,               -- punkt bazowy ukladu wspolrzednych
  frm_ox      ,               -- punkt osi OX ukladu wspolrzednych
  frm_xy      ,               -- punkt na dodatniej cwiartce ukl wsp
  place       : POSITION      -- pozycja odlozenia detalu

  TEST_PATH   : PATH

ROUTINE ResetFrames
BEGIN
  $UFRAME   = base_frame
  $UTOOL    = base_tool
END ResetFrames

ROUTINE MoveTo(midl,dest : POSITION; mt,tt, dt : INTEGER; spd : REAL)
BEGIN
  IF mt<>CIRCULAR THEN
    WITH $MOTYPE=mt,
         $TERMTYPE=tt,
         $DECELTOL=dt,
         $SPEED=spd,
    MOVE TO dest
  ELSE
    WITH $MOTYPE=mt,
         $TERMTYPE=tt,
         $DECELTOL=dt,
         $SPEED=spd,
    MOVE TO dest VIA midl
  ENDIF
END MoveTo
ROUTINE J(dest : POSITION; tt, dt : INTEGER; spd : REAL)
BEGIN
  MoveTo(dest,dest,JOINT,tt,dt,spd)
END J

ROUTINE L(dest : POSITION; tt, dt : INTEGER; spd : REAL)
BEGIN
  MoveTo(dest,dest,LINEAR,tt,dt,spd)
END L

ROUTINE C(midl,dest : POSITION; tt, dt : INTEGER; spd : REAL)
BEGIN
  MoveTo(midl,dest,CIRCULAR,tt,dt,spd)
END C
```

```
ROUTINE Circle(cntr : POSITION; r, spd : REAL)
BEGIN
  L(cntr:POS(r,0,-50,0,0,0,'N'),NODECEL,99,500)
  L(cntr:POS(r,0,0,0,0,0,'N'),FINE,1,200)
  C(cntr:POS(0,r,0,0,0,0,'N'),
    cntr:POS(-r,0,0,0,0,0,'N'),VARDECEL,80,spd)
  C(cntr:POS(0,-r,0,0,0,0,'N'),
    cntr:POS(r,0,0,0,0,0,'N'),FINE,1,spd)
  L(cntr:POS(r,0,-50,0,0,0,'N'),NODECEL,99,200)
END Circle

ROUTINE Rectangle(corn : POSITION; x,y, spd : REAL)
BEGIN
  L(corn:POS(0,0,-50,0,0,0,'N'),NODECEL,99,500)
  L(corn,FINE,1,200)
  L(corn:POS(x,0,0,0,0,0,'N'),FINE,1,spd)
  L(corn:POS(x,y,0,0,0,0,'N'),FINE,1,spd)
  L(corn:POS(0,y,0,0,0,0,'N'),FINE,1,spd)
  L(corn,FINE,1,spd)
  L(corn:POS(0,0,-50,0,0,0,'N'),NODECEL,99,200)
END Rectangle

BEGIN
  base_frame   = POS(0,0,0,0,0,0,'N')
  base_tool    = POS(0,0,0,0,0,0,'0')
  table_frame  = POS(1860,-1140,290,-180,-82.2,-49,'N')
  lab_tool     = POS(145,-55,295,-8,38,-27,'n')
  offset       = POS(-200,-100,0,0,0,0,'n')
  dest_pt1     = pick:offset
  offset       = POS(-300,-300,0,0,0,0,'n')
  dest_pt2     = pick:offset         -- obliczenie pozycji punktu 2
  offset       = POS(-100,-600,0,0,0,0,'n')
  dest_pt3     = pick:offset         -- obliczenie pozycji punktu 3
  offset       = POS(-400,-700,0,0,0,0,'n')
  dest_pt4     = pick:offset         -- obliczenie pozycji punktu 4

  $SPEED    = 150
  $TERMTYPE = NODECEL
  $MOTYPE   = JOINT

  ResetFrames

  WRITE TPMENU (CR,'TO HOME               ')
  WAIT FOR PIN[1]
  J(home,FINE,1,150)             -- przesun do punktu: home

  $UTOOL    = lab_tool
  $UFRAME   = table_frame

  WRITE TPMENU (CR,'START                 ')
  WAIT FOR PIN[1]                -- czeka na klawisz F1 na TP w User Menu
  J(start,FINE,1,150)            -- przesun do punktu: start
  J(clean_tool:POS(0,0,-50,0,0,0,'N'),NODECEL,99,800)
  L(clean_tool,FINE,1,200)       -- przesun do punktu: clean_tool
  L(clean_tool:POS(0,0,-100,0,0,0,'N'),NODECEL,99,200)
  J(pick:POS(0,0,-50,0,0,0,'N'),NODECEL,99,800)
  L(pick,FINE,1,200)             -- przesun do punktu: pick
  L(pick:POS(0,0,-20,0,0,0,'N'),NODECEL,99,200)
  J(dest_pt1:POS(0,0,-50,0,0,0,'N'),NODECEL,99,800)
  L(dest_pt1,FINE,1,200)         -- przesun do punktu: dest_pt1
  J(dest_pt1:POS(0,0,-20,0,0,0,'N'),NODECEL,99,200)
  J(dest_pt2:POS(0,0,-100,0,0,0,'N'),NODECEL,99,300)
  L(dest_pt2,FINE,1,200)         -- przesun do punktu: dest_pt2
  Circle(dest_pt2,50,100)        -- kolo o srodku dest_pt2 i promieniu 50mm
  J(dest_pt2:POS(0,0,-100,0,0,0,'N'),NODECEL,99,300)
  J(dest_pt3:POS(0,0,-50,0,0,0,'N'),NODECEL,99,800)
  L(dest_pt3,FINE,1,200)         -- przesun do punktu: dest_pt3
  Circle(dest_pt3,75,100)        -- kolo o srodku dest_pt3 i promieniu 75mm
  J(dest_pt3:POS(0,0,-20,0,0,0,'N'),NODECEL,99,200)
  J(dest_pt4:POS(0,0,-50,0,0,0,'N'),NODECEL,99,800)
  L(dest_pt4,FINE,1,200)         -- przesun do punktu: dest_pt4
  J(dest_pt4:POS(0,0,-200,0,0,0,'N'),NODECEL,99,800)
  Rectangle(dest_pt4,50,100,100) -- prost. o narozn. dest_pt4 i rozm. 50x100
  J(place:POS(0,0,-100,0,0,0,'N'),NODECEL,99,800)
  L(place,FINE,1,200)            -- przesun do punktu:place
  L(place:POS(0,0,-100,0,0,0,'N'),NODECEL,99,200)
  J(start,NODECEL,99,800)        -- przesun dp punktu: start

  ResetFrames                    -- wywolanie procedury ResetFrames

  J(home:POS(0,0,-100,0,0,0,'N'),NODECEL,99,800)
  L(home,FINE,1,200)
END lab1
```

5. PODSUMOWANIE

Według GUS już cztery piąte polskich przedsiębiorstw korzysta z robotów przemysłowych. Prognozy wskazują, że roboty przemysłowe mogą stać się coraz bardziej powszechne w polskich przedsiębiorstwach, przede wszystkich na coraz silniejszą potrzebę konkurencyjności na rynku [9]. Coraz większy nacisk kładzie się również, na jakość wytwarzanych produktów. Z roku na rok obserwowany jest spadek cen robotów, co przy wciąż rosnących pensjach pracowników jest argumentem za instalacją tego typu urządzeń (Rys. 5.1.) [10]. Najbliższy rok będzie kontynuacją trendu wzrostowego obserwowanego na rynku polskim przez ostatnie lata.



Rys. 5.1. Przyczyny wzrostu popularności robotów (według dostawców) [10]

Liczba robotów przemysłowych w Polsce w porównaniu z krajami Europy Zachodniej w dalszym ciągu jest na bardzo niskim poziomie. Współczynnik gęstości robotów w Polsce jest szacowany, jako kilka sztuk na 10000 mieszkańców. W najbardziej zrobotyzowanym kraju Europy – Niemczech – współczynnik wynosi ok. 190 [10]. Widzimy więc, że potencjał rozwoju na rynku polskim jest ogromny, a więc w przyszłości będą poszukiwani integratorzy zintegrowanych zrobotyzowanych systemów [6]. Zrobotyzowany System z udziałem Robota Przemysłowego FANUC S-420F umożliwia przećwiczenie praktycznych aspektów programowania ruchów robota przygotowywując przyszłych integratorów do pracy.

PROGRAMMING IN THE LANGUAGE KAREL WORK FANUC S-420F STRUKTURA PROGRAM - PROCEDURE

Abstract

The robotized system was introduced in the article in the support about the Work Fanuc S-420F., which systems makes possible the practical aspects of programming of the robot industrial preparation to the work in the part of the integrator integrated industrial.

BIBLIOGRAFIA

1. *Enhanced KAREL Operations Manual*. ©GMFanuc Robotics Corporation, 1991.
2. *FANUC Robot series R-J3iB Controller Line Tracking Operator's Manual*. ©FANUC, 2002.

3. Honczarenko J.: *Roboty przemysłowe*. Wydawnictwo Naukowo Techniczne, Warszawa 2004.
4. *KAREL Reference Manual*. ©GMFanuc Robotics Corporation, 1991.
5. Kuśmińska-Fijałkowska A., Łukasik Z.: *Zrobotyzowany laboratoryjny system w oparciu o robota FANUC S- 420F*. Konferencja Naukowa TransComp, Zakopane 2010.
6. Kuśmińska-Fijałkowska A., Łukasik Z.: *Struktura, tworzenie i testowanie programu zintegrowanego zrobotyzowanego systemu Fanuc S-420F*. XV Międzynarodowa Konferencja - Komputerowe Systemy Wspomagania Nauki, Przemysłu Transportu, TransComp 2011, Zakopane 2011, Logistyka 6/2011.
7. *Integracja Zrobotyzowanych Systemów Przemysłowych*. Opracowanie w Zakładzie Automatyzacji Procesów Politechnika Radomska, Radom 2008.
8. Czasopismo *Control Engineering Polska*, Nr 1(64) styczeń-luty 2010, ISSN 1731-5301.
9. www.controlengineering.pl
10. www.fanucrobotics.com
11. www.wnp.pl

Autorzy:
prof. dr hab. inż. Zbigniew Łukasik– Uniwersytet Technologiczno-Humanistyczny im. Kazimierza Pułaskiego w Radomiu
dr inż. Aldona KUŚMIŃSKA-FIJAŁKOWSKA –
Uniwersytet Technologiczno-Humanistyczny im. Kazimierza Pułaskiego w Radomiu